

Automatic skin lesion segmentation with fully convolutional-deconvolutional networks

March 16, 2017

Yading Yuan, Ming Chao and Yeh-Chi Lo
Department of Radiation Oncology
Icahn School of Medicine at Mount Sinai
New York, NY 10029
yading.yuan@mssm.edu

Abstract

This paper summarizes our method and validation results for the ISBI Challenge 2017 - Skin Lesion Analysis Towards Melanoma Detection - Part 1: Lesion Segmentation.

I. INTRODUCTION

Automatically segmenting melanoma from the surrounding skin is an essential step in computerized analysis of dermoscopic images. However, this task is not trivial because melanoma usually has a large variety of appearance in size, shape, and color along with different types of skin and texture. Meanwhile, some lesions have irregular and fuzzy borders, and in some cases the contrast between lesion and the surrounding skin is low. In addition, artifacts and intrinsic cutaneous features, such as hairs, frames, blood vessels and air bubbles can make the automatic segmentation more challenging. Researchers have developed various computer algorithms to conquer these challenges, and a recent survey can be found in [1].

We proposed a framework based on deep fully convolutional-deconvolutional neural networks (CDNN) [2, 3] to automatically segment skin lesions in dermoscopic images. Instead of developing sophisticated pre- and post-processing algorithms and hand-crafted features, we focus on designing appropriate network architecture and effective training strategies such that our deep learning model can handle images under various acquisition conditions.

II. MATERIALS AND METHODS

A. Database

We participated the part I of ISBI Challenge 2017 - Skin Lesion Analysis Towards Melanoma Detection: Lesion Segmentation. The training dataset includes 2000 dermoscopic images in .jpg format and the corresponding lesion masks in .png format. The images are of various dimensions. The lesion types involved include nevus, seborrheic keratosis and malignant melanoma. The goal is to produce accurate binary masks of various skin lesions against a variety of background. Besides training set, the organizers provide a validation dataset that includes 150 images. The participants can submit the binary masks of these 150 images and evaluate the segmentation performance online. Additional test dataset with 600 images is provided for final evaluation. The final rank is based on Jaccard index.

B. Our Approach

1. Architecture

We train a CDNN to map from input dermoscopic image to a posterior probability map. The network contains 29 layers with about 5M trainable parameters. Table I describes the architectural details. We fixed the stride as 1 and use Rectified Linear Units (ReLUs) as the activation function for each convolutional/deconvolutional layer. For output layer, we use sigmoid as the activation function. Pixel-wise classification is performed and CDNN is essentially served as a filter that projects the entire input image to a map where each element represents the probability that the corresponding input pixel belongs to the tumor. In order to address the conflict between multi-scale information aggregation and full-resolution pixel-wise classification, we implement a strategy of using upsampling and deconvolutional layers to recover lost resolution while carrying over the global perspective from pooling layers [3]. Batch normalization is added to the output of every convolutional/deconvolutional layer to reduce the internal covariate shift.

2. Pre-processing

A simple pre-processing is employed to facilitate the following learning procedure while preserving the original image information. Besides the original RGB channels, we also include the three channels in Hue-Saturation-Value color space, as well as the L channel (lightness) in CIELAB space. Each channel is rescaled to $[0, 1]$. By observing most of images in the training set have a height-to-width ratio of 3 : 4, we resize the images to 192×256 .

3. Training

We train the network using Adam optimization [4] with batch size of 16 to adjust the learning rate based on the first and the second-order moments of the gradient at each iteration. The initial learning rate is set as 0.003. In order to reduce overfitting, we use dropout with $p = 0.5$ before

Table I: Architectural details of the proposed CDNN model (Abbreviations: conv: convolutional layer; pool: max-pooling layer; decv: deconvolutional layer, ups: upsampling layer).

Conv	Filter size	No. of features	Deconv	Filter size	No. of features
conv-1-1	3×3	16	decv-1	3×3	256
conv-1-2	3×3	32	ups-1	2×2	256
pool-1	2×2	32	decv-2-1	3×3	256
conv-2-1	3×3	64	decv-2-2	3×3	128
conv-2-2	3×3	64	ups-2	2×2	128
pool-2	2×2	64	decv-3-1	4×4	128
conv-3-1	3×3	128	decv-3-2	3×3	128
conv-3-2	4×4	128	ups-3	2×2	128
pool-3	2×2	128	decv-4-1	3×3	64
conv-4-1	3×3	256	decv-4-2	3×3	32
conv-4-2	3×3	256	ups-4	2×2	32
pool-4	2×2	256	decv-5-1	3×3	16
conv-5	3×3	512	output	3×3	1

conv-4-1 and decv-5-1 in Table I, and employ two types of image augmentations to further improve the robustness of the proposed model under a wide variety of image acquisition conditions. One consists of a series of geometric transformations, including randomly flipping, shifting, rotating as well as scaling. The other type focuses on randomly normalizing the contrast of each channels in the training images. Note that these augmentations only require little extra computation, so the transformed images are generated from the original images for every mini-batch within each iteration.

We design a loss function based on Jaccard distance in this study:

$$L_{d_J} = 1 - \frac{\sum_{i,j} (t_{ij} p_{ij})}{\sum_{i,j} t_{ij}^2 + \sum_{i,j} p_{ij}^2 - \sum_{i,j} (t_{ij} p_{ij})}, \quad (1)$$

where t_{ij} and p_{ij} are target and the output of pixel (i, j) , respectively. As compared to conventionally used cross-entropy, the proposed loss function is directly related to image segmentation task because Jaccard index is a common metric to assess medical image segmentation accuracy, especially in this challenge. Meanwhile, this loss function is well adapted to the problems with high imbalance between foreground and background classes as it doesn't require any class re-balancing.

4. Post-processing

We use a dual-thresholds method to generate a binary tumor mask from the CDNN output. A relatively high threshold ($th_H = 0.8$) is firstly applied to determine the tumor center, which is

calculated as the centroid of the region that has the largest mass among the candidates from thresholding. Then a lower threshold $th_L = 0.5$ is applied to the output map. After filling small holes with morphological dilation, the final tumor mask is determined as the region that embraces the tumor center. Finally, a bagging-type ensemble strategy is implemented to combine outputs of 6 CDNNs to further improve the image segmentation performance on the testing images.

5. Implementation

Our method was implemented with Python based on Theano [5] and Lasagne¹. The experiments were conducted on a Dell XPS 8900 desktop with Intel (R) i7-6700 3.4 GHz CPU and a GPU of Nvidia GeForce GTX 1060 with 6GB GDDR5 memory. The total number of epochs was set as 500.

III. RESULTS

Our method yielded an average Jaccard index of 0.784 on the online validation dataset.

References

1. M. E. Celebi, Q. Wen, H. Iyatomi, K. Shimizu, H. Zhou, and G. Schaefer, "A state-of-the-art survey on lesion border detection in dermoscopy images." *Dermoscopy Image Analysis*, CRC Press, 2015, pp. 97-129
2. J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation." in *Proc. CVPR 2015* (2015), pp. 3431-3440
3. H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation." In *Proc. ICCV 2015* (2015), pp. 1520-1528
4. D. Kingma and J. Ba, "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014)
5. T. D. Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, et al. "Theano: A python framework for fast computation of mathematical expressions." *arXiv preprint arXiv:1605.02688* (2016)

¹<http://github.com/Lasagne/Lasagne>